Attorney Docket No.: 002-39: CSN 8544

PATENT APPLICATION

PVDM (PACKET VOICE DATA MODULE)

GENERIC BUS PROTOCOL

Inventors: ANKUR SINGLA, a citizen of India, residing at 370 Elan Village Lane #105, San Jose, California 95134;

STEPHEN DAVIES, a citizen of the United States of America, residing at 494 Northgate Drive, San Jose, California 95111; and

DAVID FERMOR, a citizen of the United Kingdom, residing at 1491 Prince Edward Way, Sunnyvale, California 94087.

Assignee: CISCO TECHNOLOGY, INC. (A California Corporation)

Entity: LARGE

BACKGROUND OF THE INVENTION

[01] Routing platforms include a motherboard having a host processor and various slave devices such as Digital Signal Processors (DSPs), Microprocessors, Application Specific Integrated Circuits (ASICs), and Field Programmable Gate Arrays (FPGAs). Also, many motherboards include a slot for holding a Packet Voice Data Module (PVDM).

[02] Communication between the host processor and the slave devices is generally accomplished utilizing proprietary, specialized interfaces for each device. For example, some devices have proprietary interfaces, others have synchronous or non-synchronous interfaces. Additionally, direct communication between slaves without host processor intervention has not been available.

[03] Another problem has been facilitating host processor transfers to large memories controlled by the slave devices. It is not practical for the host processor to map each of these slave spaces.

[04] Accordingly, a generic bus system providing efficient communication between the host processor and slave modules, efficient memory usage, and inter-slave communication is required.

BRIEF SUMMARY OF THE INVENTION

[05] In a first embodiment of the invention, a new protocol and interface specification allows for transactions with existing and future slave devices. The protocol and interface specification allows for interaction with complex slave devices such as modems, CPUs, Microcontrollers, etc.

[06] In another embodiment of the invention, a DMA engine is provides a Master with the capability of accessing a slave using either a direct access method or an indirect access method.

[07] In another embodiment of the invention, all data is transferred between the DMA engine and mailbox registers on the slave utilizing a PVDM generic bus protocol.

[08] In another embodiment of the invention, the DMA engine provides the Master the capability of performing single word accesses to the addressable region of a slave device.

[09] In another embodiment of the invention, direct communication between slave devices in made available by the interface.

[10] In another embodiment of the invention, during startup the DMA engine negotiates with the slaves to determine a desired operating mode of communication including bus width and asynchronous/synchronous mode operation.

[11] Other features and advantages of the invention will be apparent in view of the following detailed description and appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[12] Fig. 1 is a block diagram of a DMA coupled to a PVDM utilizing an interface which is one embodiment of the invention;

[13] Fig. 2 is a block diagram depicting multiple slaves coupled to the PGBP;

[14] Fig. 3 is a flow chart depicting the steps employed by an embodiment of the invention when an indirect write transaction is performed;

[15] Fig. 4 is flow chart depicting the steps employed by an embodiment of the invention when an indirect read transaction is performed;

[16] Fig. 5 depicts a system coupled in standard synchronous mode; and

[17] Fig. 6 depicts a system coupled in source synchronous mode.

DETAILED DESCRIPTION OF THE INVENTION

[18] Reference will now be made in detail to various embodiments of the invention. Examples of these embodiments are illustrated in the accompanying drawings. While the invention will be described in conjunction with these embodiments, it will be understood that it is not intended to limit the invention to any embodiment. On the contrary, it is intended to cover alternatives, modifications, and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the various embodiments. However, the present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

[19] In one embodiment of the invention, referred to below as a PVDM n Interface that defines a Generic Bus Protocol (PGBP), a generic parallel, n-bit wide data path

communications bus is defined to allow a number of major Slave devices (DSPs, Microprocessors, Application Specific ICs/FPGAs) to be used to interface with a generic PVDM module. The bus itself has no parity or CRC hardware data integrity checking. Although, we allow for CRC/parity extensions to be provided in the messages. Higher level protocol allows a DMA engine to interface multiple master devices (Host Processor, etc) to interface directly with multiple PVDM modules through the DMA (Direct Memory Access) engine.

The embodiment being described provides the following features:

- *Ultra-High-speed source synchronous, High-speed synchronous and Low-speed asynchronous modes. Synchronous speed will vary depending on the load of the modules sharing the bus. Bus start in asynchronous mode with bus mode discovery enabled.*
- *n-bit bi-directional parallel data bus, with bus width discoverable mode*
- *Message/Frame passing interface*
- *MSI (Message Signaled Interrupts)*
- *Support for multiple masters through Message Passing interface*
- *Expandable support for 32 or more slave registers per device*

[20] Each of these features will be described in detail below.

[21] The PVDM-n Interface provides the host processor (Master device) a method of sending and receiving data from slave device(s) on the PVDM modules. This interface appears to the host as a contiguous block of memory and all address translation and master device selection is handled by the DMA engine. In the following, transfers from the DMA to slave are termed Egress transfers and transfers from the slave to the DMA are termed Ingress transactions.

[22] Fig. 1 is a block diagram depicting the PGBP interface in this embodiment. In Fig. 1 a DMA engine 10 is coupled to a slave (PVDM II) 12 by the PGPB Interface. A plurality of Masters 14 are coupled to the DMA engine 10 by a Master Interface Bus 16. As is known in the art the Master Interface Bus may be a PCI Bus, Hyper-Transport Bus, PCI-X Bus, or other buses in digital systems. In Fig. 1 only a single PVDM slot is depicted as being coupled to the PGPB Interface. However, as depicted in Fig. 2, multiple PVDM slots and slave devices can be coupled to the interface. Fig. 2 also depicts the addressable memory region 20 of the Master 14 and the addressable memory region 22 of the second slave S2.

[23] The following Table describes the pin functions of the PGBP interface:

NAME	*# PINS*	*DIRECTION*	*DESCRIPTION*

MAST_DATA[n:0]	n (16/32/etc)	I/O/Z	n-bit parallel data bus. Protocol allows this bus to be used as multiplexed Address/Data Bus to provide access to greater memory allocation within the Slave device. Pull up resistors are recommended on the DMA engine or the motherboard.
MAST_ADDR[m:0]	m	I	m-bit address bus. This address bus provides access to slave device interface registers. Most of the registers are used as Mailbox/FIFO to transfer messages between the master and the slave. If MAST_DATA[n:0] is used in multiplexed mode, then these address bus need not be present.
MAST_RDWR	1	I	Master Read Write Signal. High – Read of Slave Device Low – Write to Slave Device
MAST_DATA_STROBE	1	O	Synchronous Data Valid Enable in Synchronous Mode. Data Valid Latch Enable in Asynchronous mode. DMA Engine guarantees valid data on MAST_DATA on the falling edge of Data Strobe. DMA Engine expects to sample valid data on MAST_DATA on the rising edge of Data Strobe.
SLAVE_WAIT	1	O	Slave indication to Master that it is not ready to receive data. Whenever the signal is asserted by the Slave, the Master is obliged to wait till the signal is de-asserted before continuing read/write transaction.
SLAVE_SELECTn	n	I	Slave Select Signal from the DMA engine to Slave Device indicating slave device selection.
CLKOn	n	O	CLKO Signal. The clock output signal definition will depend on the

			negotiated configuration capabilities of the interface. The CLK signal is not required in asynchronous mode and is optional in Synchronous mode providing the external system provides the clock to each module. The CLK signal is mandatory for source-sync operation.
CLKIn	n	I	CLKI Signal. The clock input signal definition will depend on the negotiated configuration capabilities of the interface. The CLKI signal is not required in asynchronous mode or in Synchronous. The CLKI signal is mandatory for source-sync operation.
RESET	1	I	Reset Signal

[24] As depicted in Fig. 2, multiple slaves may be accessed through a single PVDM module. An interface for accessing multiple slaves through a single PVDM, denoted in this specification as a PVDM-II module, is disclosed in a copending, commonly assigned patent application entitled METHOD AND APPARATUS TO COMBINE HETEROGENEOUS HARDWARE INTERFACES FOR NEXT GENERATION PACKET VOICE MODULE DEVICES, A/N 10/725,691, filed December 2, 2003, which is hereby incorporated by reference for all purposes.

[25] Thus, for the PVDM interface with two slaves, 2 SLAVE_SELECT lines are required to select either S1A or S1B. Other slave devices can be selected utilizing a single SLAVE_SELECTn line.

[26] Each slave interfaces to the PGBP through a set of mailbox registers including a Slave RX/TX Status Register, a Slave Egress MSI (Message Signal Interrupts) Register, Slave Egress Mailbox Registers, Slave Ingress Mailbox Registers, and Slave Indirect Access Address Registers. Each of these registers will be described in more detail below.

[27] The master device can communicate with the Slave Device using 32 Mailbox Registers (expandable depending on address bits availability). These registers provide capability of fast transactions into the Memory Space of the Slave Device and also allow for reading/writing to a larger Slave Memory Space through Address Mailbox Register. To allow support for current modules that only provide 4 bit address support, the fifth address

bit is obtained from the MAST_RDWR signal. This provides slaves capability of 16 write-only and 16 read-only registers. Any register that needs to be read/write will be shadowed internally within the Slave.

[28] Messages/Frames are transferred to/from the slave devices by the DMA engine using the mailbox scheme.

[29] The following are detailed descriptions and memory maps of the registers required in a slave device to interface with the PGBP of this embodiment:

Addr	Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RSVD	Reserved	Reserved															
0x01	EMR	Egress MSI	Reserved						IRI	IWI	EFW	Reserved				EMID		
0x02	XSR	Rx/Tx Status	IMID			Reserved					IWR	IRR	ERR	ES1	ES2	IRC	IWC	IMA
0x03	IMS	Ingress Message Size	Ingress Message Size Register															
0x04	MM3	Message Mailbox 3	Ingress/Egress Message Mailbox Reg3 bits[63:48] MSB															
0x05	MM2	Message Mailbox 2	Ingress/Egress Message Mailbox Reg2 bits[47:32]															
0x06	MM1	Message Mailbox 1	Ingress/Egress Message Mailbox Reg1 bits[31:16]															
0x07	MM0	Message Mailbox 0	Ingress/Egress Message Mailbox Reg0 bits[15:0] LSB															
0x08	RSVD	Reserved	Reserved															
0x09	RSVD	Reserved	Reserved															
0x0A	RSVD	Reserved	Reserved															
0x0B	RSVD	Reserved	Reserved															
0x0C	IA0	Indirect Access Address 0	Indirect Access Address Register 0 bits[15:0] LSB															
0x0D	IA1	Indirect Access Address 1	Indirect Access Address Register 1 bits[31:16]															
0x0E	IA2	Indirect Access Address 2	Indirect Access Address Register 2 bits[47:32]															
0x0F	IA3	Indirect Access Address 3	Indirect Access Address Register 3 bits[63:48] MSB															

The Slave Egress MSI Register (EMR) is used by the master device to interrupt the slave device and perform the appropriate action based on the asserted fields.

Field	*Mask*	*Read/Write*	*DESCRIPTION*
EMID: Egress Master ID	0x0007	Write	ID of the Master that originated the Egress Message
RSVD: Reserved	0x0078	Write	Reserved for individual device implementation.
EFW: Egress Frame Written	0x0080	Write	Egress Frame has been written to Slave Device (Interrupt) 1 – (Asserted by Host) Egress Frame Written 0 – (Deasserted by Slave) Egress Frame has been serviced
IWI: Indirect	0x0100	Write	Completion of write to the Egress Message

Write Interrupt			Mailbox and Slave Indirect Access Address Registers. Initiate the Write operation within the slave device
IRI: Indirect Read Interrupt	0x0200	Write	Completion of write to the Slave Indirect Access Address Registers. Initiate the Read operation within the slave device
RSVD: Reserved	0xFC00	Write	Reserved for individual device implementation.

The Slave RX/TX Status Register (XSR) informs the status of various conditions within the Slave Device. It is a READ ONLY register from the DMA engine/Master device.

Field	***Mask***	***Read/Write***	***DESCRIPTION***
IMA: Ingress Message Available	0x0001	R	This bit informs the DMA engine/Master device that the Slave device has a message available in the internal queue. 1 (Asserted) – Ingress Data Available 0 (De-asserted) – No Ingress Data in Slave
IWC: Indirect Write Completed	0x0002	R	Completion of Indirect Write
IRC: Indirect Read Completed	0x0004	R	Completion of Indirect Read
ES2: Egress Space x2 Available	0x0008	R	Egress Space for two MAXSIZE messages (typically 1500 bytes) is available on Slave
ES1: Egress Space x1 Available	0x0010	R	Egress Space for one MAXSIZE message (typically 1500 bytes) is available on Slave
ERR: Slave Error	0x0020	R	An error has occurred on the PGBP Slave.
IRR: Indirect Read Ready	0x0040	R	Informs the DMA engine/Master device that master can initiate or complete an indirect read operation. 1 (Asserted) – Slave is Idle and Ready for Read or has completed reading the data from internal memory 0 (De-asserted) – Slave is Busy reading Data
IWR: Indirect Write Ready	0x0080	R	Informs the DMA engine/Master device that master can perform an indirect write operation or has completed a recent write. 1 (Asserted) – Slave is Idle and Ready 0 (De-asserted) – Slave is Busy completing the write
RSVD: Reserved	0x1F00	R	Reserved for individual device implementation.
IMID: Ingress Master ID	0xE000	R	ID of the PGBP Master for whom the Ingress Message is intended.

The Slave Ingress Message Size Register (IMS) is not required in this embodiment. However, they may be defined if the DMA engine/Slave require implementation of this register.

Field	***Mask***	***Read/Write***	***DESCRIPTION***
Ingress Message Size	0xFFFF	Read	Size of the Ingress Message in bytes.

The Slave Ingress/Egress Message Mailbox Registers (MM 0-3) allow the Slave Devices for up to 64 Bit operation. If the Slave is identified for 32-bit operation, the data in the registers 2 and 3 are bypassed.

Slave Ingress/Egress Message Mailbox Register 0

Field	***Mask***	***Read/Write***	***DESCRIPTION***
Data[15:0]	0xFFFF	Write	Ingress/Egress Data to Slave Device Bits 15:0

Slave Ingress/Egress Message Mailbox Register 1

Field	***Mask***	***Read/Write***	***DESCRIPTION***
Data[31:16]	0xFFFF	Write	Ingress/Egress Data to Slave Device Bits 31:16

Slave Ingress/Egress Message Mailbox Register 2

Field	***Mask***	***Read/Write***	***DESCRIPTION***
Data[47:32]	0xFFFF	Write	Ingress/Egress Data to Slave Device Bits 47:32

Slave Ingress/Egress Message Mailbox Register 3

Field	***Mask***	***Read/Write***	***DESCRIPTION***
Data[63:48]	0xFFFF	Write	Ingress/Egress Data to Slave Device Bits 63:48

The Slave Indirect Access Address Registers (IA 0-3) allow the Slave Devices for up to 64 Bit addressed operation. If the Slave is identified for 32-bit address operation, the address in the registers 2 and 3 are bypassed.

Slave Indirect Access Address Register 0

Field	*Mask*	*Read/Write*	*DESCRIPTION*
Address[15:0]	0xFFFF	Write	Address to Slave Device Bits 15:0

Slave Indirect Access Address Register 1

Field	*Mask*	*Read/Write*	*DESCRIPTION*
Address[31:16]	0xFFFF	Write	Address to Slave Device Bits 31:16

Slave Indirect Access Address Register 2

Field	*Mask*	*Read/Write*	*DESCRIPTION*
Address[47:32]	0xFFFF	Write	Address to Slave Device Bits 47:32

Slave Indirect Access Address Register 3

Field	*Mask*	*Read/Write*	*DESCRIPTION*
Address[63:48]	0xFFFF	Write	Address to Slave Device Bits 63:48

[30] The DMA engine provides the master with the capability of accessing the slave using two methods – direct method (also referred to as DMA access) and/or indirect method. The Direct method requires the DMA Engine to facilitate moving of complete packet data to/from the addressable memory region of the Master to/from a message passing interface/addressable memory region on the slave device. The indirect method requires the Master device to use DMA engine to perform single word accesses (read/write) in the addressable region of the slave device.

[31] The techniques for performing direct egress and ingress accessing will be described first. In the egress direct, a transaction is performed by the DMA engine on the behalf of the master where the data is moved from the Master Device to Slave Device. Whereas, in ingress direct, a transaction is performed by the DMA engine on the behalf of the master where the data is moved from the Slave Device to Master Device. Similarly, for Indirect transactions, messages are replaced with single word message.

[32] Egress Message Transfer - The master requests/programs the DMA engine to transfer a message to the slave device. The DMA engine polls the Slave RX/TX Status Register (Egress Space Available bits) and determines whether the slave device is ready to accept any new message. In this embodiment, the message size is application specific and is typically set to 1500 bytes. If there is enough space and the slave is ready, the DMA engine moves the message from the Master Device's addressable memory region to the Slave Device's Mailbox Registers. The transfer is performed using the physical interface operation defined in detail below.

[33] The message payload movement is done to a set of two to four (programmable) 16-bit mailbox registers called Egress Message Mailbox Registers[0..3]. The DMA engine writes data to these mailbox registers in order 0 to 1/3 (depending on programmed value) in cyclic order [0,1,2,3,0,1,2,3,....] or [0,1,0,1,0,1,....]. During the data movement from the master device to the slave device, if the slave is not ready to receive further data, it can assert SLAVE_WAIT signal to indicate to the DMA engine to wait before continuation of data transfer.

[34] Upon completion of the payload movement, the DMA engine updates the Egress MSI Register, Frame Written bit, to inform the Slave device of the completion of message transfer. Also, the DMA engine updates the ID of the master that moved the message to the slave and it asserts the Egress Interrupt Bit in the Egress MSI Register. This is to request the Slave to perform internal action on the message just transferred.

[35] Ingress Message Transfer – The master programs the DMA engine to transfer a message from the slave device whenever the slave device has a message to send. The DMA engine continuously polls the Ingress Message Available bit in the Slave RX/TX Status Register. If this bit is set, it informs the DMA engine to move the message to the master indicated by the Ingress Master ID bits (in Slave RX/TX Status Register). The DMA engine now initiates the message transfer from the slave device to master device using the physical interface operation defined below.

[36] The payload movement is done from a set of two to four (programmable) 16-bit mailbox registers called Ingress Message Mailbox Registers[0..3]. The DMA engine reads data from these mailbox registers in order 0 to 1/3 (depending on programmed value) in cyclic order [0,1,2,3,0,1,2,3,....] or [0,1,0,1,0,1,....]. If during the entire message movement, if the slave device is not ready to transfer further data, it can assert SLAVE_WAIT signal to indicate to the DMA engine to wait before continuation of data transfer. The DMA engine is required to always accept data once it initiates the message transfer.

[37] In this embodiment, the first short-word in the payload from Slave may indicate the size of the message to be transferred from the slave. However, this is not required

for the protocol to work and the ING_MESSAGE _SIZE_REG may be used by the Slave Device to indicate the transaction size.

[38] As depicted in Fig. 2, the Slave S2 has access to its own memory space S2 MEMORY. Since this memory space is not directly memory mapped within the Master's memory space, the Master may be obliged to use the indirect access to transfer data to/from the slave memory region (S2 MEMORY)

[39] Indirect Slave Memory Write - During an indirect slave memory write operation, the master reads the Indirect Write Ready bit held in the Slave RX/TX Status Register. This bit informs the master that it can perform an indirect slave memory write operation (using the DMA engine) If the Slave is ready for a transfer, the Master writes a 64-bit/32-bit address in the Indirect Access Address Register, writes the 64-bit/32-bit Data to the Egress Message Mailbox Registers[0..3]. Upon completion of the write to the mailbox registers, the master then writes to the Egress MSI Register, Indirect Write Interrupt bit to request the Slave to initiate the indirect write to the requested address within its memory region. Upon initiation of the write to the Egress MSI Register, the Slave Device will clear the Indirect Write Ready bit to indicate the Slave is performing the write. Once the write is completed, the Slave will re-assert the Indirect Write Ready bit in the Slave RX/TX Status Register.

[40] Indirect Slave Memory Read - During an indirect slave memory read operation, the master reads the Indirect Read Ready bit held in the Slave RX/TX Status Register. This bit informs the master that it can perform an indirect slave memory read operation. If the Slave is ready for a transfer, the Master writes a 64-bit/32-bit address in the Indirect Access Address Register. Upon completion of the write to the address registers, the master then writes to the Egress MSI Register, Indirect Read Interrupt bit to request the slave to initiate the indirect Read to the requested address within its memory region. Upon write to the Egress MSI Register, the slave will clear the Indirect Read Ready bit to indicate the Slave is performing the read from its internal memory map and loading the values in the Ingress Message Mailbox Registers[0..3]. Once the read is completed, the Slave will re-assert the Indirect Read Ready bit in the Slave RX/TX Status Register. The master will poll for this bit to be re-asserted, once it is ready, the master can complete the Read of the Ingress Message Mailbox Registers[0..3].

[41] Thus, as depicted in Fig. 1 the indirect access feature allows a master to transfer data between a region of main memory (A) to a region of the a desired slave memory, area (B) of S2 MEMORY without requiring the master include the slave memory in its memory space.

[42] The present embodiment provides for direct communication between slaves without host processor involvement. For example, in Fig. 2 the second slave device S2 can communicate directly with the fourth slave device S4 by programming the DMA engine to access S4 by the direct or indirect method. The second slave S2 utilizes a Master Bus Interface (not shown) to program the DMA engine.

[43] The DMA Engine provides support for Synchronous operation of the PGBP Bus and/or Asynchronous operation of the PGBP Bus. Synchronous operation is supported via Source-Synchronous clock operation or traditional synchronous operation. The PGBP interface allows for support of very high throughput through the multi-loaded PVDM-II module(s). Synchronous/Source-Sync operation allows up-to 100MHz/100MHz DDR (Double Data Rate) bus operation (depending on system architecture), providing maximum raw throughput of 1200/TBDMbps. Asynchronous operation allows interface to modules that cannot operate with minimal skew requirements of the synchronous mode. Asynchronous operation supports lower raw throughput of 300Mbps (Practical throughput is about 75% of the theoretical number).

[44] During the bootstrapping phase, the DMA engine operates in Asynchronous mode to allow handshaking with a low performance Slave device not capable of communicating in the Synchronous mode. This allows the DMA engine/the Host and the Slave to negotiate the desired operation mode – Source Synchronous, Synchronous, and/or Asynchronous. In addition, the DMA engine will negotiate the bus interface widths. The initial bus width is 8/16 (as hard configured in the DMA engine) but can be increased dynamically by negotiating with the slave device.

[45] This negotiation provides great flexibility and expandability to the PGBP and allows slave devices of different capabilities regarding data transfer speed, types of system clocks, and bus width to be coupled to the PGBP of this embodiment.

[46] Examples of hardware interfaces for implementing the Synchronous and Source Synchronous Interfaces will now be described with reference to Figs. 5 and 6.

[47] Fig. 5 depicts the traditional synchronous mode. In this mode an external system clock device can be used to distribute the clock to each end-point. The design becomes a standard multi-drop synchronous interface with end-point timing controlled via matched length clock distribution techniques.

[48] Fig. 6 depicts the source synchronous mode. In this mode the DMA Block interface is configured to have a single clock output and a single clock input per PVDM interface. The DMA block sources data to each PVDM based on a common shared master clock. Data sourced from the PVDM is received based on individual receive clock inputs (CLKIn). The MAST_DATA bus is routed to the PVDM modules with either a daisy-chain bus or STAR topology. Each CLKOn signal is individually routed to each PVDM module with a length necessary to ensure a fixed timing relationship between the write data and the CLKOn signal edge(s). The PVDM sources the CLKIn clock, the DMA Block receives separate clocks from each PVDM which are then used to receive the data into individual read FIFOs. Note that for STAR topologies the receive clock phase relationships are identical resulting in a single read FIFO implementation. This mechanism allows for completely deterministic source synchronous behavior without the need for clock gating.

[49] The invention may be implemented as hardware and/or program code, stored on a computer readable medium, that is executed by a digital computer. The computer readable medium may include, among other things, magnetic media, optical media, electro-magnetic fields encoding digital information, and so on.

[50] The invention has now been described with reference to the preferred embodiments. Alternatives and substitutions will now be apparent to persons of skill in the art. For example, the particular numbers of bus lines will vary according to the requirements of a system. Also, the polling of interrupt bits can be replace by actively interrupting the DMA engine. Accordingly, it is not intended to limit the invention except as provided by the appended claims.